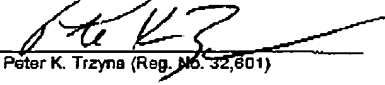


**RECEIVED
CENTRAL FAX CENTER****DEC 04 2006**

I hereby certify that this correspondence is being filed by facsimile and addressed to MS: Fee Amendment, Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date indicated below.

Date: December 4, 2006Signed: 
Peter K. Trzyna (Reg. No. 32,601)

PATENT

Paper No.

File: Proflowers-P1-01

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventors	:	David McCarter, Jared Polis
Serial No.	:	09/776,956
Filed	:	February 5, 2001
For	:	GENERATING A COURIER SHIPPING LABEL OR THE LIKE, INCLUDING AN ORNAMENTAL GRAPHIC DESIGN, AT A NON-COURIER PRINTER
Group Art Unit	:	2625
Examiner	:	PHAM, Thierry L.

MS: Fee Amendment
Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

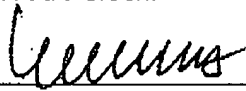
DECLARATION OF PRIOR INVENTION

S I R :

1. My name is William Strauss.
2. At a date prior to March 10, 1998, I was in charge of operations for ProFlowers, Inc. and became President and Chief Operating Officer of the company, to which the above-identified patent application was assigned.
3. I am familiar with the patent application and the patent claims.
4. This Declaration is to establish completion of the invention in the United States at a date prior to March 10, 1998, that is the effective filing date of U.S. Patent No. 5,984,778, which is cited by the Examiner in the Office Action having a mailing date of 06/02/07.

Best Available Copy

5. Under my supervision, this invention was completed and in use in the United States to make Federal Express deliveries prior to March 10, 1998.
6. Corroborating my declaration is computer code which was filed as an appendix in the patent application from which this application claims priority, i.e., Ser. No. 09/149,680 filed 09/08/1998. Attention is drawn to particularly to pages 654-675 that code, which is attached hereto. Pages 654-675 pertain to "CFX_FEDEXSHIP" marked as "Copyright 97".
7. This portion of the code pertains to ProFlowers operations that enabled the labeling printed by a supplier, as illustrated by the label attached hereto.
8. Collectively, the foregoing corroborates my declaration that this invention was completed and used in the United States prior to March 10, 1998.
9. I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statement may jeopardize the validity of the application or any patent issued thereon.



William Strauss

Date: 12/1/06

```
////////////////////////////////////
//
// CFX_FEDEXSHIP - Cold Fusion custom tag
//
// Copyright 97. All Rights Reserved.
//

#include "stdafx.h"                // Standard MFC libraries
#include "cfx.h"                   // CFX Custom Tag API
#include "WebTransaction.h"        // FedEx WebAPI

// Constants
#define TAG_ERROR_HEADER          "Error occurred in tag CFX_FedExTrack"

// Forward declarations of implementation functions
int GetLabelData(char* buffer, char labelData[]);
LPCSTR getField(char *cBuffer, const char *cFieldCode);
LPCSTR GetRequiredAttribute( CCFXRequest* pRequest, LPCSTR lpszParamName );
int BuildFedExBuffer( CCFXRequest* pRequest, WebTransaction* cWebTran,
    CString TransactionType ,
    CString TransactionID ,
    CString Reference ,
    CString ShipperName ,
    CString ShipperCompany ,
    CString ShipperAddress1 ,
    CString ShipperAddress2 ,
    CString ShipperCity ,
    CString ShipperState ,
    CString ShipperZip ,
    CString ShipperAccount ,
    CString ShipperPhone ,
    CString PayorAccount ,
    CString MeterNr ,
    CString DestinationCompanyName ,
    CString DestinationContactName ,
    CString DestinationAddress1 ,
    CString DestinationAddress2 ,
    CString DestinationCity ,
    CString DestinationState ,
    CString DestinationZip ,
    CString DestinationCountry ,
    CString DestinationPhone ,
    CString ServiceType ,
    CString ShipDate ,
    CString PaymentCode ,
    CString PackageHeight ,
    CString PackageWidth ,
    CString PackageLength ,
    CString PackageWeight ,
    CString NumberOfPackages ,
    CString SignatureRelease ,
    CString ReleaseAuthNumber ,
    CString Residential ,
    CString LabelFormat ,
    CString FutureShipDate ,
    CString SpecialServices ,
    CString LabelDirectory ,
    CString DuplicatePackage
```

- 6.54 -

```

    } ;

void ProcessTagRequest( CCFXRequest* pRequest, WebTransaction* cWebTran )
{
    try
    {
        int iFedExErr ;
        char cFedExErr[6] ;

        // Retrieve attributes passed to the tag
        CString TransactionType = GetRequiredAttribute( pRequest, "Transa
        CString TransactionID = GetRequiredAttribute( pRequest, "Transa
        CString Reference = GetRequiredAttribute( pRequest, "Reference"
        CString ShipperName = GetRequiredAttribute( pRequest, "ShipperN
        CString ShipperCompany = GetRequiredAttribute( pRequest, "Shipp
        CString ShipperAddress1 = GetRequiredAttribute( pRequest, "Ship
        CString ShipperAddress2 = GetRequiredAttribute( pRequest, "Ship
        CString ShipperCity = GetRequiredAttribute( pRequest, "ShipperC
        CString ShipperState = GetRequiredAttribute( pRequest, "Shipper
        CString ShipperZip = GetRequiredAttribute( pRequest, "ShipperZi
        CString ShipperAccount = GetRequiredAttribute( pRequest, "Shipp
        CString ShipperPhone = GetRequiredAttribute( pRequest, "Shipper
        CString PayorAccount = GetRequiredAttribute( pRequest, "PayorAc
        CString MeterNr = GetRequiredAttribute( pRequest, "MeterNr" ) ;
        CString DestinationCompanyName = GetRequiredAttribute( pRequest
        CString DestinationContactName = GetRequiredAttribute( pRequest
        CString DestinationAddress1 = GetRequiredAttribute( pRequest, "
        CString DestinationAddress2 = GetRequiredAttribute( pRequest, "
        CString DestinationCity = GetRequiredAttribute( pRequest, "Dest
        CString DestinationState = GetRequiredAttribute( pRequest, "Des
        CString DestinationZip = GetRequiredAttribute( pRequest, "Desti
        CString DestinationCountry = GetRequiredAttribute( pRequest, "D
        CString DestinationPhone = GetRequiredAttribute( pRequest, "Des
        CString ServiceType = GetRequiredAttribute( pRequest, "ServiceT
        CString ShipDate = GetRequiredAttribute( pRequest, "ShipDate" )
        CString PaymentCode = GetRequiredAttribute( pRequest, "PaymentC
        CString PackageHeight = GetRequiredAttribute( pRequest, "Packag
        CString PackageWidth = GetRequiredAttribute( pRequest, "Package
        CString PackageLength = GetRequiredAttribute( pRequest, "Packag
        CString PackageWeight = GetRequiredAttribute( pRequest, "Packag
        CString NumberOfPackages = GetRequiredAttribute( pRequest, "Num
        CString SignatureRelease = GetRequiredAttribute( pRequest, "Sig
        CString ReleaseAuthNumber = GetRequiredAttribute( pRequest, "Re
        CString Residential = GetRequiredAttribute( pRequest, "Resident
        CString LabelFormat = GetRequiredAttribute( pRequest, "LabelFor
        CString FutureShipDate = GetRequiredAttribute( pRequest, "Futur
        CString SpecialServices = GetRequiredAttribute( pRequest, "Spec
        CString LabelDirectory = GetRequiredAttribute( pRequest, "Label
        CString DuplicatePackage = GetRequiredAttribute( pRequest, "Dup

        // default output fields
        pRequest->SetVariable( "ErrorCode", "" ) ;
        pRequest->SetVariable( "ErrorMsg", "" ) ;
        pRequest->SetVariable( "TrackingNr", "" ) ;
        pRequest->SetVariable( "DeliveryDay", "" ) ;

        iFedExErr = BuildFedExBuffer( pRequest, cWebTran,
                                     TransactionType ,

```

```
TransactionID ,
Reference ,
ShipperName ,
ShipperCompany ,
ShipperAddress1 ,
ShipperAddress2 ,
ShipperCity ,
ShipperState ,
ShipperZip ,
ShipperAccount ,
ShipperPhone ,
PayorAccount ,
MeterNr ,
DestinationCompanyName ,
DestinationContactName ,
DestinationAddress1 ,
DestinationAddress2 ,
DestinationCity ,
DestinationState ,
DestinationZip ,
DestinationCountry ,
DestinationPhone ,
ServiceType ,
ShipDate ,
PaymentCode ,
PackageHeight ,
PackageWidth ,
PackageLength ,
PackageWeight ,
NumberOfPackages ,
SignatureRelease ,
ReleaseAuthNumber ,
Residential ,
LabelFormat ,
FutureShipDate ,
SpecialServices ,
LabelDirectory ,
DuplicatePackage
) ;
_itoa( iFedExErr, cFedExErr, 10 );
pRequest->SetVariable( "ConnectErr", cFedExErr ) ;

// Output optional debug info
if ( pRequest->Debug() )
{
    pRequest->WriteDebug( "Debug info..." ) ;
}

// Catch Cold Fusion exceptions & re-raise them
catch( CCFXException* e )
{
    pRequest->ReThrowException( e ) ;
}

// Catch ALL other exceptions and throw them as
// Cold Fusion exceptions (DO NOT REMOVE! --
// this prevents the server from crashing in
```

- 656 -

```

    // case of an unexpected exception)
    catch( ... )
    {
        pRequest->ThrowException(
            "Error occurred in tag CFX_FEDEXSHIP",
            "Unexpected error occurred while processing tag." );
    }
}

```

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//      BuildFedExBuffer
//      builds buffer, connects to and disconnects from
//      FedEx WEBAPI server over the Internet
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
int BuildFedExBuffer( CCFXRequest* pRequest, WebTransaction* cWebTran,
    CString TransactionType ,
    CString TransactionID ,
    CString Reference ,
    CString ShipperName ,
    CString ShipperCompany ,
    CString ShipperAddress1 ,
    CString ShipperAddress2 ,
    CString ShipperCity ,
    CString ShipperState ,
    CString ShipperZip ,
    CString ShipperAccount ,
    CString ShipperPhone ,
    CString PayorAccount ,
    CString MeterNr ,
    CString DestinationCompanyName ,
    CString DestinationContactName ,
    CString DestinationAddress1 ,
    CString DestinationAddress2 ,
    CString DestinationCity ,
    CString DestinationState ,
    CString DestinationZip ,
    CString DestinationCountry ,
    CString DestinationPhone ,
    CString ServiceType ,
    CString ShipDate ,
    CString PaymentCode ,
    CString PackageHeight ,
    CString PackageWidth ,
    CString PackageLength ,
    CString PackageWeight ,
    CString NumberOfPackages ,
    CString SignatureRelease ,
    CString ReleaseAuthNumber ,
    CString Residential ,
    CString LabelFormat ,
    CString FutureShipDate ,
    CString SpecialServices ,
    CString LabelDirectory ,
    CString DuplicatePackage
)

WebTransaction xctn("", 0, "", "");
int iLastError, iLabelSize ;
CString sReturnField, TrackingNr ; 4657-

```

```

char LabelFileName[256] ;
FILE *fLabelFile ;

if ( TransactionType == "international" )
{
    xctn.AddVar("0", "51", true);
    xctn.AddVar("26", GetRequiredAttribute(pRequest, "intlCustomsVal");
    xctn.AddVar("68", GetRequiredAttribute(pRequest, "intlCurrency");
    xctn.AddVar("69", GetRequiredAttribute(pRequest, "intlCarriageVal");
    xctn.AddVar("70", GetRequiredAttribute(pRequest, "intlDutyTaxBil");
    xctn.AddVar("71", GetRequiredAttribute(pRequest, "intlDutyTaxBil");
    xctn.AddVar("74", GetRequiredAttribute(pRequest, "intlCountryOfU");
    xctn.AddVar("75", GetRequiredAttribute(pRequest, "intlWeightType");
    xctn.AddVar("77", GetRequiredAttribute(pRequest, "intlUnitWeight");
    xctn.AddVar("79", GetRequiredAttribute(pRequest, "intlDescription");
    xctn.AddVar("80", GetRequiredAttribute(pRequest, "intlCountryOfM");
    xctn.AddVar("117", GetRequiredAttribute(pRequest, "intlShipperCo");
    xctn.AddVar("127", GetRequiredAttribute(pRequest, "intlUnitValue");
    xctn.AddVar("401", GetRequiredAttribute(pRequest, "intlQuantity");
    xctn.AddVar("414", GetRequiredAttribute(pRequest, "intlUnitOfMea");
    xctn.AddVar("1116", GetRequiredAttribute(pRequest, "intlDimUnits");
}
else
{
    // default is domestic shipment
    xctn.AddVar("0", "1", true);
}
//xctn.AddVar("1", cWebTran->GetNextTransID());
xctn.AddVar("1", TransactionID);
xctn.AddVar("25", Reference);
xctn.AddVar("32", ShipperName);
xctn.AddVar("4", ShipperCompany);
xctn.AddVar("5", ShipperAddress1);
xctn.AddVar("6", ShipperAddress2);
xctn.AddVar("7", ShipperCity);
xctn.AddVar("8", ShipperState);
xctn.AddVar("9", ShipperZip);
xctn.AddVar("10", ShipperAccount);
xctn.AddVar("183", ShipperPhone);
xctn.AddVar("20", PayorAccount);
xctn.AddVar("498", MeterNr);
xctn.AddVar("11", DestinationCompanyName);
xctn.AddVar("12", DestinationContactName);
xctn.AddVar("13", DestinationAddress1);
xctn.AddVar("14", DestinationAddress2);
xctn.AddVar("15", DestinationCity);
xctn.AddVar("16", DestinationState);
xctn.AddVar("17", DestinationZip);
xctn.AddVar("50", DestinationCountry);
xctn.AddVar("18", DestinationPhone);
xctn.AddVar("22", ServiceType);
xctn.AddVar("24", ShipDate);
xctn.AddVar("23", PaymentCode);
xctn.AddVar("57", PackageHeight);
xctn.AddVar("58", PackageWidth);
xctn.AddVar("59", PackageLength);
xctn.AddVar("21", PackageWeight);
xctn.AddVar("116", NumberOfPackages);
xctn.AddVar("51", SignatureRelease);
xctn.AddVar("1118", ReleaseAuthNumber); - 658 -

```

```

xctn.AddVar("440", Residential);
xctn.AddVar("187", LabelFormat);
xctn.AddVar("1119", FutureShipDate);
xctn.AddVar("39", SpecialServices);
xctn.AddVar("408", DuplicatePackage);
xctn.AddVar("99", "");

// write send buffer to file
/*
FILE *fptrl;
SetCurrentDirectory("f:\\bmao\\misc\\");
fptrl = fopen("lastsend.txt", "w+");
if (fptrl != NULL)
{
    fputs(xctn.m_OutBuffer, fptrl);
    fclose(fptrl);
}
*/
// end write

int returnlen;
iLastError = xctn.Connect();
if (iLastError != WEBAPI_OK)
{
    pRequest->SetVariable( "ConnectErrMsg", xctn.GetErrorSt);
    return iLastError;
}

memset(xctn.m_InBuffer, 0, WEBAPI_MAX_BUFFER_SIZE);
iLastError = WEBAPITransaction(xctn.m_OutBuffer, strlen(xctn.m_
    WEBAPI_MAX_BUFFER_SIZE, &returnlen);
if (iLastError != WEBAPI_OK)
{
    pRequest->SetVariable( "ConnectErrMsg", xctn.GetErrorSt);
    return iLastError - 10000;
}

xctn.m_bDecoded = FALSE;
iLastError = xctn.Disconnect();
pRequest->SetVariable( "ConnectErrMsg", xctn.GetErrorString() )

// write returned buffer to file; MUST BE IN BINARY MODE (b)!!
/*
fptrl = fopen("lastrecv.txt", "wb");
if ( fptrl != NULL)
{
    fputs(xctn.m_InBuffer, fptrl);
    fclose(fptrl);
}
*/
// end write

// get error code
sReturnField = getField(xctn.m_InBuffer, "\"2,\"");
if (sReturnField == "")
{
    sReturnField = "0";
}
pRequest->SetVariable( "ErrorCode", sReturnField );

```

- 459 -


```

// get error message
sReturnField = getField(xctn.m_InBuffer, "\"3,\"");
pRequest->SetVariable( "ErrorMsg", sReturnField );

// get tracking number
TrackingNr = getField(xctn.m_InBuffer, "\"29,\"");
pRequest->SetVariable( "TrackingNr", TrackingNr );

// get delivery date
sReturnField = getField(xctn.m_InBuffer, "\"194,\"");
pRequest->SetVariable( "DeliveryDay", sReturnField );

// get label and write output file
char labelData[WEBAPI_MAX_BUFFER_SIZE];
//xctn.GetLabelData(labelData);
iLabelSize = GetLabelData(xctn.m_InBuffer, labelData);
if ( iLabelSize > 0 )
{
    // write label to file
    SetCurrentDirectory(LabelDirectory);
    strcpy(LabelFileName, TrackingNr);
    strcat(LabelFileName, ".gif");
    fLabelFile = fopen(LabelFileName, "wb");
    if ( fLabelFile != NULL )
    {
        int numwritten = fwrite( labelData, sizeof( char), iLabelSize, fLabelFile );
        fclose(fLabelFile);
    }
    // end write
}

return iLastError;
}

int GetLabelData(char* buffer, char labelData[])
{
    int j = 0;
    int q;
    int count = 0;
    char cchar;
    char temp[2];
    char tempVarName[100];
    char hexTable[20];
    int btofs = 0;
    BOOL bLabel = FALSE;

    tempVarName[0] = '\0';
    labelData[0] = '\0';

    strcpy(hexTable, "0123456789ABCDEF");

    for (UINT i = 0; i < strlen(buffer); i++)
    {
        switch (buffer[i])
        {
            case 0:
            {
                break;
            }

```

```

    }
    case 34:
    {
        if (j)
        {
            if (bLabel)
            {
                return count;
            }
            j = 0;
            tempVarName[0] = '\0';
        }
        else
        {
            if (strcmp(tempVarName, "188") == 0)
            {
                bLabel = TRUE;
            }
            j = 1;
        }
        break;
    }
    default:
    {
        if (j)
        {
            if (bLabel)
            {
                switch (btofs)
                {
                    case 1:
                    {
                        cchar = 0;
                        for (q = 0; q <
                        {
                            if (tou
                            {
                                cchar =
                                break;
                            }
                        }
                        btofs = 2;
                        break;
                    }
                    case 2:
                    {
                        for (q = 0; q <
                        {
                            if (tou
                            {
                                cchar =
                                break;
                            }
                        }
                        labelData[count
                        count++;
                    }
                }
            }
        }
    }

```

- 661 -

```

        btofs = 0;
        break;
    }
    default:
    {
        if (buffer[i] ==
        {
            btofs =
        }
        else
        {
            labelDa
            count++
        }
        break;
    }
}
}
}
else
{
    if (buffer[i] != ',')
    {
        temp[0] = buffer[i];
        temp[1] = 0;
        strcat(tempVarName, temp);
    }
    break;
}
}
return count;
}
}

```

```

// Parse the returned buffer for the contents of a specified field
LPCSTR getField(char *cBuffer, const char *cFieldCode)
{

```

```

    char tmpString[2048] ;
    UINT iStart, iLen, iLenFieldCode ;
    char *pdest, *pdest1 ;
    LPSTR ptmpString ;

```

```

    tmpString[0] = '\0' ;
    iLenFieldCode = strlen(cFieldCode) ;

```

```

    pdest = strstr( cBuffer, cFieldCode );
    if (pdest == NULL)
    {
        tmpString[0] = '\0' ;
        ptmpString = &tmpString[0] ;
        return ptmpString ;
    }

```

```

    iStart = pdest - cBuffer + iLenFieldCode ; - 662 -

```

```
pdest1 = strstr( _strninc(cBuffer, iStart), "\"" );
iLen = pdest1 - pdest - iLenFieldCode ;
strncpy( tmpString, pdest+iLenFieldCode, iLen );
tmpString[iLen] = '\\0' ;
```

```
ptmpString = &tmpString[0] ;
return ptmpString ;
}
```

```
// Get the value for the passed attribute (throw an exception
// if the attribute was not passed to the tag)
LPCSTR GetRequiredAttribute( CCFXRequest* pRequest, LPCSTR lpszAttribName )
{
    // Verify that the attribute exists (throw an exception
    // if it does not)
    if ( !pRequest->AttributeExists(lpszAttribName) )
    {
        CString strErr =
            "The required attribute " + CString(lpszAttribName) +
            " was not passed to the tag. " ;
        pRequest->ThrowException( TAG_ERROR_HEADER, strErr ) ;
    }

    // Return the attribute
    return pRequest->GetAttribute( lpszAttribName ) ;
}
```

- 663 -

```

////////////////////////////////////
//
// CFX_FEDEXTRACK - Cold Fusion custom tag
//
// Copyright 97. All Rights Reserved.
//

#include "stdafx.h" // Standard MFC libraries
#include "cfx.h" // CFX Custom Tag API
#include "WebTransaction.h" // FedEx WebAPI

// Constants
#define TAG_ERROR_HEADER "Error occurred in tag CFX_FedExTrack"

// Variables

// Forward declarations of implementation functions
LPCSTR getField(char *cBuffer, const char *cFieldCode) ;
LPCSTR GetRequiredAttribute( CCFXRequest* pRequest, LPCSTR lpszParamName ) ;
int BuildFedExBuffer(CCFXRequest* pRequest, WebTransaction* cWebTran, CString, (
void ProcessTagRequest( CCFXRequest* pRequest, WebTransaction* cWebTran )
{
    try
    {
        int iFedExTrackErr ;
        char cFedExTrackErr[6] ;

        // Retrieve attributes passed to the tag
        CString strTrackingNr = GetRequiredAttribute( pRequest, "Tracki
        CString strDestCountry = GetRequiredAttribute( pRequest, "DestC
        CString strShipDate = GetRequiredAttribute( pRequest, "ShipDate

        // default output fields
        pRequest->SetVariable( "ErrorCode", "" ) ;
        pRequest->SetVariable( "ErrorMsg", "" ) ;
        pRequest->SetVariable( "DeliveryDate", "" ) ;
        pRequest->SetVariable( "DeliveryTime", "" ) ;
        pRequest->SetVariable( "DeliveryTo", "" ) ;
        pRequest->SetVariable( "SignedFor", "" ) ;
        pRequest->SetVariable( "DispatchException", "" ) ;
        pRequest->SetVariable( "ServiceType", "" ) ;
        pRequest->SetVariable( "PackageType", "" ) ;
        pRequest->SetVariable( "ScanActivity", "" ) ;

        iFedExTrackErr = 3 ;
        BuildFedExBuffer( pRequest, cWebTran, strTrackingNr, strDestCou
        _itoa( iFedExTrackErr, cFedExTrackErr, 10 ) ;
        pRequest->SetVariable( "ConnectErr", cFedExTrackErr ) ;

        // Output optional debug info
        if ( pRequest->Debug() )
        {
            pRequest->WriteDebug( "Debug info..." ) ;
        }
    }
}

```

- 664 -

```

// Catch Cold Fusion exceptions & re-raise them
catch( CCFXException* e )
{
    pRequest->ReThrowException( e ) ;
}

// Catch ALL other exceptions and throw them as
// Cold Fusion exceptions (DO NOT REMOVE! --
// this prevents the server from crashing in
// case of an unexpected exception)
catch( ... )
{
    pRequest->ThrowException(
        "Error occurred in tag CFX_FEDEXTRACK",
        "Unexpected error occurred while processing tag." ) ;
}
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// BuildFedExBuffer
// builds buffer, connects to and disconnects from
// FedEx WEBAPI server over the Internet
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
int BuildFedExBuffer( CCFXRequest* pRequest, WebTransaction* cWebTran, CString :
{
    WebTransaction xctn("", 0, "", "");
    int iLastError ;
    int nScanActivity, iPos, i, j ;
    CString sReturnField ;
    char cFieldCode[8] ;
    char tmpString[71] ;
    char ScanActivity[1200] ; // up to 15 fields, 70 + 1 chars each

    xctn.AddVar("0", "402", true);
    xctn.AddVar("1", cWebTran->GetNextTransID());
    xctn.AddVar("29", strTrackingNr);
    xctn.AddVar("50", strDestCountry);
    xctn.AddVar("24", strShipDate);
    xctn.AddVar("99", "");

    // write send buffer to file
    /*
    FILE *fptrl;
    SetCurrentDirectory("f:\\bmao\\misc\\");
    fptrl = fopen("lastsend.txt", "w+");
    if (fptrl != NULL)
    {
        fputs(xctn.m_OutBuffer, fptrl);
        fclose(fptrl);
    }
    */
    // end write

    int returnlen;
    iLastError = xctn.Connect() ;
    if (iLastError != WEBAPI_OK)
    {
        - 66.5 -
    }
}

```

```

        pRequest->SetVariable( "ConnectErrMsg", xctn.GetErrorStr
        return iLastError;
    }
    memset(xctn.m_InBuffer, 0, WEBAPI_MAX_BUFFER_SIZE);
    iLastError = WEBAPITransaction(xctn.m_OutBuffer, strlen(xctn.m_C
        WEBAPI_MAX_BUFFER_SIZE, &returnlen);
    iLastError = xctn.Disconnect();
    pRequest->SetVariable( "ConnectErrMsg", xctn.GetErrorString() )

// write returned buffer to file
/*
    fptr1 = fopen("lastrecv.txt", "w+");
    if ( fptr1 != NULL)
    {
        fputs(xctn.m_InBuffer, fptr1);
        fclose(fptr1);
    }
*/
// end write

// get error code
    sReturnField = getField(xctn.m_InBuffer, "\"2,\"");
    if (sReturnField == "")
    {
        sReturnField = "0" ;
    }
    pRequest->SetVariable( "ErrorCode", sReturnField ) ;

// get error message
    sReturnField = getField(xctn.m_InBuffer, "\"3,\"");
    pRequest->SetVariable( "ErrorMsg", sReturnField ) ;

// get delivery date field and set Cold Fusion variable
    sReturnField = getField(xctn.m_InBuffer, "\"1720,\"");
    pRequest->SetVariable( "DeliveryDate", sReturnField ) ;

// get delivery time field and set Cold Fusion variable
    sReturnField = getField(xctn.m_InBuffer, "\"1707,\"");
    pRequest->SetVariable( "DeliveryTime", sReturnField ) ;

// get Delivery To field and set Cold Fusion variable
    sReturnField = getField(xctn.m_InBuffer, "\"1705,\"");
    pRequest->SetVariable( "DeliveryTo", sReturnField ) ;

// get Signed For field and set Cold Fusion variable
    sReturnField = getField(xctn.m_InBuffer, "\"1706,\"");
    pRequest->SetVariable( "SignedFor", sReturnField ) ;

// get Dispatch Exception field and set Cold Fusion variable
    sReturnField = getField(xctn.m_InBuffer, "\"1709,\"");
    pRequest->SetVariable( "DispatchException", sReturnField ) ;

// get Service Type field and set Cold Fusion variable
    sReturnField = getField(xctn.m_InBuffer, "\"1704,\"");
    pRequest->SetVariable( "ServiceType", sReturnField ) ;

// get Package Type field and set Cold Fusion variable
    sReturnField = getField(xctn.m_InBuffer, "\"1718,\"");
    pRequest->SetVariable( "PackageType", sReturnField ) ;

```

- 666 -

```

// get scan activity fields and set Cold Fusion variable
sReturnField = getField(xctn.m_InBuffer, "\"1715,\"") ;
nScanActivity = atoi( sReturnField ) ;

iPos = 0 ;
for (i=1; i<=nScanActivity; i++)
{
    _itoa( 1720+i, cFieldCode, 10 ) ;

    for (j=4; j>=0; j--)
    {
        cFieldCode[j+1] = cFieldCode[j] ;
    }
    cFieldCode[0] = '\"' ;
    cFieldCode[5] = ',' ;
    cFieldCode[6] = '\"' ;
    cFieldCode[7] = '\0' ;

    sReturnField = getField(xctn.m_InBuffer, cFieldCode) ;
    strcpy( tmpString, sReturnField ) ;
    for (j=0; j<strlen(sReturnField);j++)
    {
        ScanActivity[iPos] = tmpString[j] ;
        iPos++ ;
    }

    ScanActivity[iPos] = '<' ;
    iPos++ ;
    ScanActivity[iPos] = 'b' ;
    iPos++ ;
    ScanActivity[iPos] = 'r' ;
    iPos++ ;
    ScanActivity[iPos] = '>' ;
    iPos++ ;
}

ScanActivity[iPos] = '\0' ;
pRequest->SetVariable( "ScanActivity", ScanActivity ) ;

return iLastError;
}

```

```

// Parse the returned buffer for the contents of a specified field
LPCSTR getField(char *cBuffer, const char *cFieldCode)
{
    char tmpString[2048] ;
    UINT iStart, iLen, iLenFieldCode ;
    char *pdest, *pdest1 ;
    LPSTR ptmpString ;

    tmpString[0] = '\0' ;
    iLenFieldCode = strlen(cFieldCode) ;

    pdest = strstr( cBuffer, cFieldCode ) ;
    if (pdest == NULL)
    {
        tmpString[0] = '\0' ;
        ptmpString = &tmpString[0] ;
    }
}

```

- (66 7) -


```
        return ptmpString ;
    }

    iStart = pdest - cBuffer + iLenFieldCode ;
    pdest1 = strstr( _strninc(cBuffer, iStart), "\"" );
    iLen = pdest1 - pdest - iLenFieldCode ;
    strncpy( tmpString, pdest+iLenFieldCode, iLen );
    tmpString[iLen] = '\0' ;

    ptmpString = &tmpString[0] ;
    return ptmpString ;
}

// Get the value for the passed attribute (throw an exception
// if the attribute was not passed to the tag)
LPCSTR GetRequiredAttribute( CCFXRequest* pRequest, LPCSTR lpszAttribName )
{
    // Verify that the attribute exists (throw an exception
    // if it does not)
    if ( !pRequest->AttributeExists(lpszAttribName) )
    {
        CString strErr =
            "The required attribute " + CString(lpszAttribName) +
            " was not passed to the tag. " ;
        pRequest->ThrowException( TAG_ERROR_HEADER, strErr ) ;
    }

    // Return the attribute
    return pRequest->GetAttribute( lpszAttribName ) ;
}
```

- 648 -

```

////////////////////////////////////
//
// CFX_FEDEXTRACKADV - Cold Fusion custom tag
//
// Copyright 98. All Rights Reserved.
//

#include "stdafx.h"          // Standard MFC libraries
#include "cfx.h"             // CFX Custom Tag API
#include "WebTransaction.h"   // FedEx WebAPI

// Constants
#define TAG_ERROR_HEADER     "Error occurred in tag CFX_FedExTrackAdv"

// Variables

// Forward declarations of implementation functions
LPCSTR getField(char *cBuffer, const char *cFieldCode) ;
LPCSTR GetRequiredAttribute( CCFXRequest* pRequest, LPCSTR lpszParamName ) ;
int BuildFedExBuffer(CCFXRequest* pRequest, WebTransaction* cWebTran, CString, C

void ProcessTagRequest( CCFXRequest* pRequest, WebTransaction* cWebTran )
{
    try
    {
        int iFedExTrackErr ;
        char cFedExTrackErr[6] ;

        // Retrieve attributes passed to the tag
        CString strTrackingNr = GetRequiredAttribute( pRequest, "TrackingNr" ) ;
        CString strAccountNr = GetRequiredAttribute( pRequest, "ShipperAccountNr" ) ;
        CString strShipDate = GetRequiredAttribute( pRequest, "ShipDate" ) ;

        // default output fields
        pRequest->SetVariable( "ErrorCode", "" ) ;
        pRequest->SetVariable( "ErrorMsg", "" ) ;
        pRequest->SetVariable( "DeliveryDate", "" ) ;
        pRequest->SetVariable( "DeliveryTime", "" ) ;
        pRequest->SetVariable( "DeliveryTo", "" ) ;
        pRequest->SetVariable( "SignedFor", "" ) ;
        pRequest->SetVariable( "DispatchException", "" ) ;
        pRequest->SetVariable( "ServiceType", "" ) ;
        pRequest->SetVariable( "PackageType", "" ) ;
        pRequest->SetVariable( "ScanActivity", "" ) ;

        iFedExTrackErr = 3 ;
        BuildFedExBuffer( pRequest, cWebTran, strTrackingNr, strAccountNr, strShipDate,
            _itoa( iFedExTrackErr, cFedExTrackErr, 10 ) ;
        pRequest->SetVariable( "ConnectErr", cFedExTrackErr ) ;

        // Output optional debug info
        if ( pRequest->Debug() )
        {
            pRequest->WriteDebug( "Debug info..." ) ;
        }
    }
}

```

-669-

```

// Catch Cold Fusion exceptions & re-raise them
catch( CCFXException* e )
{
    pRequest->ReThrowException( e ) ;
}

// Catch ALL other exceptions and throw them as
// Cold Fusion exceptions (DO NOT REMOVE! --
// this prevents the server from crashing in
// case of an unexpected exception)
catch( ... )
{
    pRequest->ThrowException(
        "Error occurred in tag CFX_FEDEXTRACKADV",
        "Unexpected error occurred while processing tag." ) ;
}
}

```

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//      BuildFedExBuffer
//      builds buffer, connects to and disconnects from
//      FedEx WEBAPI server over the Internet
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
int BuildFedExBuffer( CCFXRequest* pRequest, WebTransaction* cWebTran, CString :
{
    WebTransaction xctn("", 0, "", "");
    int iLastError ;
    int nScanActivity, iPos, i, j ;
    CString sReturnField ;
    char cFieldCode[8] ;
    char tmpString[71] ;
    char ScanActivity[1200] ;      // up to 15 fields, 70 + 1 chars each

    xctn.AddVar("0", "403", true);
    xctn.AddVar("1", cWebTran->GetNextTransID());
    xctn.AddVar("29", strTrackingNr);
    xctn.AddVar("10", strAccountNr);
    xctn.AddVar("24", strShipDate);
    xctn.AddVar("99", "");

    // write send buffer to file
    /*
    FILE *fptrl;
    SetCurrentDirectory("f:\\bmao\\misc\\");
    fptrl = fopen("lastsend.txt", "w+");
    if (fptrl != NULL)
    {
        fputs(xctn.m_OutBuffer, fptrl);
        fclose(fptrl);
    }
    */
    // end write

    int returnlen;
    iLastError = xctn.Connect() ;
    if (iLastError != WEBAPI_OK)
    {

```

- 670 -

```

        pRequest->SetVariable( "ConnectErrMsg", xctn.GetErrorStr);
        return iLastError;
    }
    memset(xctn.m_InBuffer, 0, WEBAPI_MAX_BUFFER_SIZE);
    iLastError = WEBAPITransaction(xctn.m_OutBuffer, strlen(xctn.m_
        WEBAPI_MAX_BUFFER_SIZE, &returnlen);
    iLastError = xctn.Disconnect();
    pRequest->SetVariable( "ConnectErrMsg", xctn.GetErrorString() )

// write returned buffer to file
/*
    fptr1 = fopen("lastrecv.txt", "w+");
    if ( fptr1 != NULL)
    {
        fputs(xctn.m_InBuffer, fptr1);
        fclose(fptr1);
    }
*/
// end write

// get error code
sReturnField = getField(xctn.m_InBuffer, "\"2,\"");
if (sReturnField == "")
{
    sReturnField = "0" ;
}
pRequest->SetVariable( "ErrorCode", sReturnField ) ;

// get error message
sReturnField = getField(xctn.m_InBuffer, "\"3,\"");
pRequest->SetVariable( "ErrorMsg", sReturnField ) ;

// get delivery date field and set Cold Fusion variable
sReturnField = getField(xctn.m_InBuffer, "\"1720,\"");
pRequest->SetVariable( "DeliveryDate", sReturnField ) ;

// get delivery time field and set Cold Fusion variable
sReturnField = getField(xctn.m_InBuffer, "\"1707,\"");
pRequest->SetVariable( "DeliveryTime", sReturnField ) ;

// get Delivery To field and set Cold Fusion variable
sReturnField = getField(xctn.m_InBuffer, "\"1705,\"");
pRequest->SetVariable( "DeliveryTo", sReturnField ) ;

// get Signed For field and set Cold Fusion variable
sReturnField = getField(xctn.m_InBuffer, "\"1706,\"");
pRequest->SetVariable( "SignedFor", sReturnField ) ;

// get Dispatch Exception field and set Cold Fusion variable
sReturnField = getField(xctn.m_InBuffer, "\"1709,\"");
pRequest->SetVariable( "DispatchException", sReturnField ) ;

// get Service Type field and set Cold Fusion variable
sReturnField = getField(xctn.m_InBuffer, "\"1704,\"");
pRequest->SetVariable( "ServiceType", sReturnField ) ;

// get Package Type field and set Cold Fusion variable
sReturnField = getField(xctn.m_InBuffer, "\"1718,\"");
pRequest->SetVariable( "PackageType", sReturnField ) ;

```

- 671 -

```

// get scan activity fields and set Cold Fusion variable
sReturnField = getField(xctn.m_InBuffer, "\"1715,\"");
nScanActivity = atoi( sReturnField );

iPos = 0 ;
for (i=1; i<=nScanActivity; i++)
{
    _itoa( 1720+i, cFieldCode, 10 );

    for (j=4; j>=0; j--)
    {
        cFieldCode[j+1] = cFieldCode[j] ;
    }
    cFieldCode[0] = '\"' ;
    cFieldCode[5] = ',' ;
    cFieldCode[6] = '\"' ;
    cFieldCode[7] = '\0' ;

    sReturnField = getField(xctn.m_InBuffer, cFieldCode) ;
    strcpy( tmpString, sReturnField );
    for (j=0; j<strlen(sReturnField); j++)
    {
        ScanActivity[iPos] = tmpString[j] ;
        iPos++ ;
    }

    ScanActivity[iPos] = '<' ;
    iPos++ ;
    ScanActivity[iPos] = 'b' ;
    iPos++ ;
    ScanActivity[iPos] = 'r' ;
    iPos++ ;
    ScanActivity[iPos] = '>' ;
    iPos++ ;
}

ScanActivity[iPos] = '\0' ;
pRequest->SetVariable( "ScanActivity", ScanActivity ) ;

return iLastError;
}

```

```

// Parse the returned buffer for the contents of a specified field
LPCSTR getField(char *cBuffer, const char *cFieldCode)
{

```

```

    char tmpString[2048] ;
    UINT iStart, iLen, iLenFieldCode ;
    char *pdest, *pdest1 ;
    LPSTR ptmpString ;

    tmpString[0] = '\0' ;
    iLenFieldCode = strlen(cFieldCode) ;

    pdest = strstr( cBuffer, cFieldCode );
    if (pdest == NULL)
    {
        tmpString[0] = '\0' ;
        ptmpString = &tmpString[0] ;

```

- 672 -

```
        return ptmpString ;
    }

    iStart = pdest - cBuffer + iLenFieldCode ;
    pdest1 = strstr( _strninc(cBuffer, iStart), "\"" );
    iLen = pdest1 - pdest - iLenFieldCode ;
    strncpy( tmpString, pdest+iLenFieldCode, iLen );
    tmpString[iLen] = '\0' ;

    ptmpString = &tmpString[0] ;
    return ptmpString ;
}

// Get the value for the passed attribute (throw an exception
// if the attribute was not passed to the tag)
LPCSTR GetRequiredAttribute( CCFXRequest* pRequest, LPCSTR lpszAttribName )
{
    // Verify that the attribute exists (throw an exception
    // if it does not)
    if ( !pRequest->AttributeExists(lpszAttribName) )
    {
        CString strErr =
            "The required attribute " + CString(lpszAttribName) +
            " was not passed to the tag. " ;
        pRequest->ThrowException( TAG_ERROR_HEADER, strErr ) ;
    }

    // Return the attribute
    return pRequest->GetAttribute( lpszAttribName ) ;
}
```

- 673 -

```

////////////////////////////////////
//
// CFX_NOTIFYSUPPLIERPS - Cold Fusion custom tag
//
// Copyright 98. All Rights Reserved.
//

#include "stdafx.h"          // Standard MFC libraries
#include "cfx.h"             // CFX Custom Tag API

// Constants
#define TAG_ERROR_HEADER    "Error occurred in tag CFX_FedExTrack"

// Forward declarations of implementation functions
LPCSTR GetRequiredAttribute( CCFXRequest* pRequest, LPCSTR lpszParamName );

void ProcessTagRequest( CCFXRequest* pRequest )
{
    try
    {
        const unsigned TokenMax = 25 ; // allows for 25 tokens, incl.
        LPCSTR TokenID[TokenMax+1], TokenValue[TokenMax+1] ; // not
        FILE *pFile ;
        char *pdest ;
        const unsigned MAX_BUFFER_SIZE = 5120 ; // 5K
        char InBuffer[MAX_BUFFER_SIZE], OutBuffer[MAX_BUFFER_SIZE] ;
        char ErrorMessage[256] ;
        unsigned long int ErrorCode, iFLenIn, iPos, i, j,
            CurrentPosInBuffer, CurrentPosOutBuffer ;
        //char cDebug[6] ; // debug use only

        // default output fields
        pRequest->SetVariable( "ErrorCode", "0" ) ;
        pRequest->SetVariable( "ErrorMsg", "" ) ;

        // Retrieve attributes passed to the tag
        TokenID[1] = "#productName#" ;
        TokenValue[1] = GetRequiredAttribute( pRequest, "productName" ) ;
        TokenID[2] = "#productCode#" ;
        TokenValue[2] = GetRequiredAttribute( pRequest, "productCode" ) ;
        TokenID[3] = "#orderID#" ;
        TokenValue[3] = GetRequiredAttribute( pRequest, "orderID" ) ;
        TokenValue[4] = GetRequiredAttribute( pRequest, "sheetNr" ) ;
        TokenID[4] = "#sheetNr#" ;
        TokenID[5] = "#quantity#" ;
        TokenValue[5] = GetRequiredAttribute( pRequest, "quantity" ) ;
        TokenID[6] = TokenID[5] ;
        TokenValue[6] = TokenValue[5] ;
        TokenID[7] = "#orderDate#" ;
        TokenValue[7] = GetRequiredAttribute( pRequest, "orderDate" ) ;
        TokenID[8] = "#deliveryDate#" ;
        TokenValue[8] = GetRequiredAttribute( pRequest, "deliveryDate" ) ;
        TokenID[9] = "#recipientFirstName#" ;
        TokenValue[9] = GetRequiredAttribute( pRequest, "recipientFirst" ) ;
        TokenID[10] = "#recipientLastName#" ;
        TokenValue[10] = GetRequiredAttribute( pRequest, "recipientLast" ) ;
        TokenID[11] = "#companyName#" ;
    }
}

```

- 674 -

```

TokenValue[11] = GetRequiredAttribute( pRequest, "companyName" )
TokenID[12] = "#address1#";
TokenValue[12] = GetRequiredAttribute( pRequest, "address1" );
TokenID[13] = "#address2#";
TokenValue[13] = GetRequiredAttribute( pRequest, "address2" );
TokenID[14] = "#city#";
TokenValue[14] = GetRequiredAttribute( pRequest, "city" );
TokenID[15] = "#state#";
TokenValue[15] = GetRequiredAttribute( pRequest, "state" );
TokenID[16] = "#zip#";
TokenValue[16] = GetRequiredAttribute( pRequest, "zip" );
TokenID[17] = "#countryCode#";
TokenValue[17] = GetRequiredAttribute( pRequest, "countryCode" );
TokenID[18] = "#phone1#";
TokenValue[18] = GetRequiredAttribute( pRequest, "phone1" );
TokenID[19] = "#phone2#";
TokenValue[19] = GetRequiredAttribute( pRequest, "phone2" );
TokenID[20] = "#cardMsg1" + "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX#";
TokenValue[20] = GetRequiredAttribute( pRequest, "cardMsg1" );
TokenID[21] = "#cardMsg2" + "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX#";
TokenValue[21] = GetRequiredAttribute( pRequest, "cardMsg2" );
TokenID[22] = "#cardMsg3" + "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX#";
TokenValue[22] = GetRequiredAttribute( pRequest, "cardMsg3" );
TokenID[23] = "#cardMsg4" + "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX#";
TokenValue[23] = GetRequiredAttribute( pRequest, "cardMsg4" );
TokenID[24] = "#cardMsg5" + "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX#";
TokenValue[24] = GetRequiredAttribute( pRequest, "cardMsg5" );
TokenID[25] = "showpage";
TokenValue[25] = "";

```

```

CString Template = GetRequiredAttribute( pRequest, "Template" );
CString OutputDir = GetRequiredAttribute( pRequest, "OutputDir" );
CString OutputFileName = GetRequiredAttribute( pRequest, "Output" );

```

```

ErrorCode = 0;

```

```

// debug file

```

```

/*

```

```

SetCurrentDirectory( OutputDir );

```

```

FILE *fDebug;

```

```

fDebug = fopen( "debug.txt", "w+" );

```

```

*/

```

```

// end debug file

```

```

pFile = fopen( Template, "rb" );

```

```

if ( pFile == NULL )

```

```

{

```

```

    strcpy( ErrorMessage, "error opening file " );

```

```

    strcat( ErrorMessage, Template );

```

```

    pRequest->SetVariable( "ErrorCode", "1" );

```

```

    pRequest->SetVariable( "ErrorMsg", ErrorMessage );

```

```

}

```

```

else

```

```

{

```

```

    iFlenIn = fread( &InBuffer, sizeof( char ), MAX_BUFFER,
    fclose( pFile );

```

```

    CurrentPosInBuffer = 0;

```

```

    CurrentPosOutBuffer = 0;

```

- 675 -

ORDER: Blue Irides [201R35 ST]

Order Nr.: JSCHU8113300 (Sheet 1 of 1)

Quantity: 1

Order Date: 23-Apr-98

Delivery By: 24-Apr-98

Ship To: Nicholas Rodgers
Proflowers, Inc.
8438 S. Yarrow St.
Littleton, CO 80123 US
Phone: 303-585-5555

Dear Nik,
I hope you enjoy these Irides! This is one of
the many assortments we offer.
~Jared

SHIPPER'S FLINK ACCOUNT NUMBER



FROM:

Proflowers.com
830 Mendenhall

Redwood City CA 94060

TO:
NICHOLAS RODGERS
PROFLOWERS, INC.
8438 S. YARROW ST.

LITTLETON CO 80123

REF: JSCHU8113300 (1 of 1)



TRK # 7800 7883 4884 PRIORITY OVERNIGHT

FORM ID: 0201

80123-CO-US

DEN

WN BJC

FedEx.

Federal Express

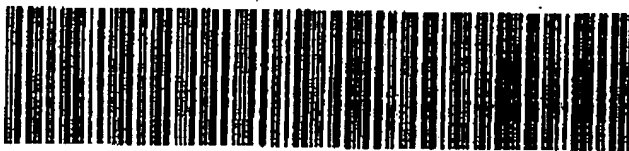
SHIP DATE: 23APR98
SHIP WT: 3.1 LB
DIMENSIONS: 14 X 9 X 6

RELEASED:
6208951

FRI

A1

SHIP DATE: 24APR98



**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☒ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.